НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

«До захисту допущено»

Завідувач кафедри

_____ Бурау Н. І.

«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра

за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Промисловий робот. Розробка алгоритму керування маніпулятором.»

Виконав:
студент ІІІ курсу, групи ПГ-п61
Бондаренко Михайло Володимирович _____

Керівник:
ст. викл. Мураховський С.А.

_____

Консультант з:

_____

Рецензент:
ст. викл. Божко К. М.

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Анотація

Змістом дипломного проекту являється описання методики створення алгоритму керування маніпулятором. В першому розділі приводиться опис історії розвитку робототехніки та аналіз основних функціональних елементів промислового робота. Також приводить опис розповсюджених алгоритмів побудови руху маніпулятора, та приводиться алгоритм їх використання. Другий розділ повністю присвячений кінематиці, а саме опис методики Денавіта-Хартенберга, постановка та вирішення прямої та зворотної задач кінематики. Останній розділ присвячений найбільш важливій функціональній складовій промислового робота – методиці створення алгоритму, приведені два можливі шляхи реалізації поставленої задачі: використання середовищ Arduino та Labview.

Ключові слова: промисловий робот, алгоритм, Arduino, Labview, кінематика, метод Денавіта-Хартенберга

Аннотация

Содержанием дипломного проекта является описание методики создания алгоритма управления манипулятором. В первом разделе приводится описание истории развития робототехники и анализ основных функциональных элементов промышленного робота. Также приводит описание распространенных алгоритмов построения движения манипулятора, и приводится алгоритм их использования. Второй раздел полностью посвящен кинематике, а именно описание методики Денавита-Хартенберга, постановка и решение прямой и обратной задач кинематики. Последний раздел посвящен наиболее важной функциональной составляющей промышленного робота - методике создания алгоритма, приведены два возможных пути реализации поставленной задачи: использование сред Arduino и Labview.

Ключевые слова: промышленный робот, алгоритм, Arduino, Labview, кинематика, метод Денавита-Хартенберга

Annotation

The content of the diploma project is a description of the methodology for creating a manipulator control algorithm. The first section gives a description of the history about development of robotics and analysis of the main functional elements of the industrial robot. Also describes the widespread algorithm for constructing the motion of the manipulator, and gives a method for their use. The second section is entirely devoted to kinematics, namely the description of Denavi-Hartenberg's method, the formulation and solution of the direct and reciprocal problems of kinematics. The last section is devoted to the most important functional component of an industrial robot - the methodology for creating an algorithm, two possible ways of realizing the task: using Arduino and Labview environments.

Keywords: industrial robot, algorithm, Arduino, Labview, kinematics, Denavi-Hartenberg method.

Зміст

Вступ

Однією з основних рушійних сил автоматизації сучасного виробництва є промислові роботи-маніпулятори. Їх розробка і впровадження дозволили вийти підприємствам на новий науково-технічний рівень виконання завдань, перерозподілити обов'язки між технікою і людиною, підвищити продуктивність.

Ідея створення приладів, котрі будуть автоматизовувати окремі діяльності людини, бере свій початок з глибокої давнини. Людина завжди прагнула створити штучних механічних прислуг, котрі мали б усі переваги людини і в той же час, не страждали від недоліків, що притаманні живим організмам.

Сама по собі робототехніка, як наукова галузь, являється досить молодою, оскільки формально була сформована у середині XX ст. Однак вже на етапі зародження до неї було приковано увагу багатьох спеціалістів, науково-дослідницьких та конструкторських організацій різноманітних галузей промислової діяльності.

Така тенденція інтересу не являється безпідставною і обумовлена, здебільшого, гуманним бажанням звільнити людину від праці у екстремальних, або навіть небезпечних умовах праці. Слід також звернути увагу на факт того, що людина, як соціальна істота, не може нормально працювати без дотримання ряду життєво важливих факторів.

Не менш важливим фактором являється розробка програмного забезпечення промислового робота, адже без нього, робот, сам по собі, не представляє із себе нічого вартого, оскільки без програми він не виконує нічого. Програмування їх руху, здебільшого, здійснюється по методиці навчання, а програма, в свою чергу, передбачає запис відповідних рухів маніпулятора.

Найбільш ефективним застосуванням промислових роботів являється автоматизація транспортних, допоміжних та деяких технологічних операцій в умова малосерійного та серійного виробництва.

Розділ 1. Оглядова частина

1.1. Промислові роботи та маніпулятори

Промисловий робот - призначений для виконання рухових і керуючих функцій у виробничому процесі маніпуляційний робот, тобто автоматичний пристрій, що складається з маніпулятора і перепрограмованого пристрою керування, що формує керуючі впливу, які задають необхідний рух виконавчих органів маніпулятора. Застосовується для переміщення предметів виробництва та виконання різних технологічних операцій [1].

Альтернативно можна привести визначення промислового робота, взяте з ГОСТ 25686-85 – "автоматична машина, стаціонарна або пересувна, що складається з виконавчого пристрою у вигляді маніпулятора, має кілька ступенів рухомості, і перепрограмованого пристрою програмного управління для виконання у виробничому процесі рухових і керуючих функцій" [2].

Використання промислових роботів дозволяє провести процес автоматизації виробництва – це процес у розвитку машинного виробництва, при якому функції управління і контролю, що раніше виконувалися людиною, передаються приладам і автоматичним пристроям. Введення автоматизації на виробництві дозволяє значно підвищити продуктивність праці, забезпечити стабільну якість продукції, що випускається, скоротити частку робочих, зайнятих в різних сферах виробництва.

Для того щоб краще зрозуміти місце і наукові завдання створення промислових роботів, доцільно розглянути загальну постановку проблеми «робототехніка» і так звані три покоління роботів. В даний час у всій спеціальній науковій та популярній літературі прийнято поділ роботів на три покоління. Незважаючи на те що цей розподіл дещо умовний, такий прийом дозволяє чіткіше зазначити особливості різних типів роботів і якимось чином класифікувати їх [3].

І покоління - промислові роботи. По думці, що склалася, промислові роботи (ПР) являють собою автоматичні пристрої, оснащені однією або кількома «руками». Рух руки промислового робота здійснюється за кількома керованим координатам (від двох до

восьми) із заданою програмованої швидкістю і необхідною точністю. На кінці руки монтують кисть з робочим органом. Переміщення робочого органу відбувається в межах зони обслуговування ПР.

Найважливішою відмінною рисою промислових роботів є те, що вони, як правило, не мають датчиків зворотного зв'язку і не можуть реагувати на зміни зовнішнього середовища. Передбачається, що середовище, що оточує робота, строго організоване, детерміноване і незмінне. Ця особливість дещо обмежує області застосування промислових роботів. Програмування їх рухів здійснюється в основному методом навчання, а програма передбачає запис всіх рухів маніпулятора.

II покоління - адаптивні роботи. Адаптивні роботи, тобто роботи, керовані пристроєм адаптивного управління, відносяться до більш досконалих роботів і, як це випливає з назви, можуть реагувати на зміни зовнішнього середовища. Вони оснащені датчиками зворотного зв'язку - сенсорними пристроями, тобто наділені можливостями відчувати. Ця особливість і є головною, що відрізняє адаптивні роботи від роботів I покоління. Можливість корегувати програму в залежності від зміни параметрів зовнішнього середовища дозволяє істотно розширити сферу застосування роботів цього покоління в порівнянні з промисловими роботами. Маніпулятори робочих органів адаптивних роботів не мають принципових відмінностей від таких у роботів I покоління.

III покоління - роботи зі штучним інтелектом. Роботи III покоління, найбільш складні і досконалі, раніше називали інтегральними. Слід зазначити, що зараз вони знаходяться в стадії розробки. Створені окремі, як правило експериментальні та дослідні, зразки, а застосування їх в промисловості тільки починається. Конструкція роботів з штучним інтелектом істотно відрізняється від раніше описаних тим, що вони дуже часто виконуються рухомими, оснащуються колісним або гусеничним ходом. Досить поширеним науковим напрямком є створення крокуючих роботів, а також роботів, призначених для досліджень космосу і океану.

Роботи зі штучним інтелектом оснащуються потужними ЕОМ і в цілому складніше і дорожче роботів II покоління. Математичне забезпечення інтегральних

роботів досить складне. В пам'ять робота потрібно занести математичну модель зовнішнього середовища і загальну мету, якої необхідно досягти. Конкретна програма дій виробляється в процесі руху робота на підставі зіставлення моделі зовнішнього середовища, загальної мети та інформації, одержаної від органів відчуттів.

Розглянемо більш детально функціональну схему промислового робота, а саме проведемо аналіз елементів які до неї входять. У складі промислового робота є механічна частина (що включає один або кілька маніпуляторів) і система управління цією механічною частиною. Крім цього, робот може мати засоби відчуттів (утворюють в сукупності інформаційно-сенсорну систему), сигнали від яких надходять до системи управління [4].

Маніпулятор - це керований механізм (або машина), який призначений для виконання рухових функцій, аналогічних функціям руки людину при переміщенні об'єктів в просторі, і оснащений робочим органом. У деяких випадках до складу промислового робота можуть входити два (або більше число) маніпуляторів[5].

Виконавчий механізм маніпулятора, як правило, являє собою відкритий кінематичний ланцюг, ланки якої послідовно з'єднані між собою зчленуваннями різного типу; в переважній більшості випадків, однак, зустрічаються кінематичні пари V-го класу (що володіють одним ступенем рухливості), а серед останніх - поступальні і обертальні зчленування [6].

Поєднання і взаємне розташування ланок і зчленувань визначає число ступенів рухливості, а також область дії маніпуляційної системи робота. Зазвичай припускають, що перші три зчленування у виконавчому механізмі маніпулятора реалізують транспортні (або переносні) ступеня рухливості (забезпечуючи висновок робочого органу в заданий місце), а інші реалізують орієнтують ступеня рухливості (відповідаючи за потрібну орієнтацію робочого органу) [7]. Залежно від виду перших трьох зчленувань більшість роботів відносять до однієї з чотирьох категорій:

- роботи, що працюють в декартовій системі координат - роботи, у яких всі три початкових зчленування є поступальними

• роботи, що працюють в циліндричній системі координат - роботи, у яких серед початкових сполучень два поступальних і одне обертальний

• роботи, що працюють в сферичної системі координат - роботи, у яких серед початкових сполучень одне поступальних і два обертальних

• роботи, що працюють в кутовий, або обертальної, системі координат - роботи, у яких всі три початкових зчленування є обертовими

Для деяких маніпуляторів підрозділ ступенів рухливості на переносні і орієнтують не прийнято. Прикладом можуть служити маніпулятори з кінематичною надмірністю (тобто з числом ступенів рухливості, більше ніж шість); тут управління переміщенням робочого органу і управління його орієнтацією не “розв'язані” за окремими групами зчленувань [7].

У деяких випадках маніпулятор промислового робота встановлюють на рухому основу, що означає наділення його додатковими ступенями рухливості. Так, маніпулятор встановлюють на рейки або ж на рухому каретку, що пересувається по підлогової колії або уздовж підвісних напрямних [8].

На кінці маніпулятора (на його «зап'ясті») розташовується робочий орган - пристрій, призначений для виконання спеціального завдання. В якості робочого органу може виступати захватний пристрій або технологічний інструмент.

Найбільш універсальним різновидом пристрою захоплення є схват - пристрій, в якому захоплення і утримання об'єкта проводяться за допомогою відносного переміщення частин даного пристрою. Як правило, схват за своєю конструкцією нагадує кисть людської руки: захоплення об'єкта здійснюється за допомогою механічних «пальців». Для захоплення плоских предметів використовуються захватні пристрої з пневматичної присоскою. Застосовують також гаки (для підняття деталей з конвеєрів), черпаки або совки (для рідких, сипучих або гранульованих речовин). Для захоплення ж безлічі однотипних деталей застосовують спеціалізовані конструкції (наприклад, магнітні захватні пристрої) [9].

За способом утримання об'єкта захватні пристрої поділяють на [6]:

• схоплюючі (механічні тужавіння і пристрої з еластичними робочими камерами, в які нагнітають рідину або стиснене повітря);

• підтримуючі (в них об'єкт не затискають, а застосовують для його утримання нижню поверхню, виступаючі частини об'єкта або наявні в ньому отвори);

• утримуючі (в них на об'єкт здійснюють силовий вплив за рахунок різних фізичних ефектів: вакуумні, магнітні і електростатичні захоплення, адгезія і т. п.).

Для приведення ланок маніпулятора і пристрої схвата в рух використовують електричні, гідравлічні або пневматичні приводи. Гідравлічні приводи кращі у випадках, коли треба забезпечити значну величину зусиль, що розвиваються або високу швидкодію; зазвичай такими приводами забезпечуються масивні роботи великої вантажопідйомності. Електричні приводи не володіють настільки ж великою силою або швидкодією, але дозволяють домогтися кращих характеристик точності. Пневматичні приводи зазвичай застосовують для невеликих за розмірами роботів, які виконують прості і швидкі циклічні операції. За наявними оцінками, приблизно в 50% сучасних промислових роботах використовується електричний привід, в 30% - гідравлічний і в 20% - пневматичний [10].

Застосування роботів в промисловому виробництві має ряд переваг, зокрема [11]:

• підвищення продуктивності праці (оскільки відкривається можливість використання технологічного обладнання в три-чотири зміни і 365 днів на рік);

• зменшення витрат виробництва і підвищення конкурентоспроможності;

• раціональне використання обладнання і виробничих приміщень;

• поліпшення якості продукції, пов'язане з підвищенням точності виконання технологічних операцій;

• виключення впливу людського фактора на конвеєрних виробництвах, а також при проведенні монотонних робіт, що вимагають високої точності;

• виключення впливу на персонал шкідливих чинників, характерних для виробництв з підвищеною небезпекою;

• зниження термінів окупності інвестицій.

1.2. Алгоритм та система управління маніпулятора

Алгоритм – кінцева сукупність точно заданих правил рішення довільного класу задач або набір інструкцій, що описують порядок дій виконавця для вирішення деякої задачі. Раніше замість слова «порядок» використовувалося слово «послідовність», але в міру розвитку паралельності в роботі комп'ютерів слово «послідовність» стали замінювати більш загальним словом «порядок».

У розвитку систем управління промислових роботів можна простежити два напрямки. Одне з них бере свій початок від систем програмного управління верстатами і вилилося в створення автоматично керованих промислових маніпуляторів. Друге призвело до появи напівавтоматичних біотехнічних та інтерактивних систем, в яких в управлінні діями промислового робота бере участь людина-оператор [12].

Таким чином, промислові роботи можна поділити на такі три типу (кожен з яких, в свою чергу, поділяють на кілька різновидів [13][10]:

- Автоматичні роботи:
  - Програмні роботи (роботи з програмним управлінням) – найпростіша різновид автоматично керованих промислових роботів, до цих пір широко використовуваних в силу їх дешевизну на різних промислових підприємствах для обслуговування нескладних технологічних процесів. У таких роботах відсутня сенсорна частина, а всі дії виконуються циклічно по жорсткої програмі, закладеної в пам'ять пристрою, що запам'ятовує.
  - Адаптивні роботи (роботи з адаптивним керуванням) - роботи, оснащені сенсорною частиною (системою відчуттів) і забезпечені набором програм. Сигнали, що надходять до системи управління від датчиків, аналізуються нею, і в залежності від результатів приймається рішення про подальші дії робота, яка передбачає перехід від однієї програми до іншої (зміна технологічної операції). Апаратне і програмне забезпечення - в принципі те саме, що і в попередньому випадку, але до його можливостей пред'являються підвищені вимоги.

○ Роботи що навчаються - роботи, дії яких повністю формуються в ході навчання (людина за допомогою спеціальної плати задає порядок дій робота, і цей порядок дій записується в пам'ять пристрою, що запам'ятовує).

○ Інтелектуальні роботи (роботи з елементами штучного інтелекту) - роботи, здатні за допомогою сенсорних пристроїв самостійно сприймати і розпізнавати оточення, будувати модель середовища, і автоматично приймати рішення про подальші дії, а також самонавчатися в міру накопичення власного досвіду діяльності.

• Біотехнічні роботи:

○ Командні роботи (роботи з командним управлінням) - маніпулятори, в яких людина-оператор дистанційно задає з командного пристрою рух в кожному зчленуванні (строго кажучи, це - не роботи в повному розумінні слова, а «напівроботи»).

○ Копіюючі роботи (роботи з копіюючим управлінням) - маніпулятори, що копіюють дії приводиться в рух оператором пристрою, що задає, кінематично подібного виконавчого механізму маніпулятора (як і в попередньому випадку, такі маніпулятори можна вважати «напівроботами»).

○ Напівавтоматичні роботи - роботи, при управлінні якими людина-оператор задає лише рух робочого органу маніпулятора, а формування узгоджених рухів в зчленуваннях система управління роботів здійснює самостійно.

• Інтерактивні роботи:

○ автоматизовані роботи (роботи з автоматизованим управлінням) - роботи, що чергують автоматичні режими управління з біотехнічних.

○ Супервізорні роботи (роботи з супервизорним управлінням) - роботи, що виконують автоматично всі етапи заданого циклу операцій, але здійснюють перехід від одного етапу до іншого по команді людини-оператора.

○ Діалогові роботи (роботи з діалоговим управлінням) - автоматичні роботи (будь-якого різновиду), здатні взаємодіяти з людиною-оператором,

використовуючи мову того чи іншого рівня (включаючи подачу текстових або голосових команд і відповідні повідомлення робота).

Більшість сучасних роботів функціонує на основі принципів зворотного зв'язку, підлеглого управління та ієрархічності системи управління роботом. Ієрархічна побудова системи управління роботом передбачає поділ системи управління на горизонтальні шари (рівні): на верхньому рівні здійснюється управління загальною поведінкою робота, на рівні планування рухів проводиться розрахунок необхідної траєкторії руху робочого органу, на рівні координації приводів організовується злагоджена робота приводів, що забезпечує необхідне переміщення робочого органу і, нарешті, на рівні приводу безпосередньо здійснюється управління двигуном, що відповідає за конкретну ступінь рухливості маніпулятора [7].

Сам по собі маніпулятор не буде представляти із себе нічого якщо він не матиме самого базового елемента необхідного для функціонування системи – алгоритму планування траєкторії руху. Завдання планування траєкторії в робототехніці полягає в знаходженні оптимального шляху з початкового положення в кінцеве для складних тіл в деякому просторі.

Існує кілька принципових підходів до вирішення завдання планування траєкторії. Комбінаторний підхід використовує аналітичні алгоритми, виробляє моделювання конфігураційного простору і знаходить повне рішення. Ефективне застосування комбінаторних алгоритмів можливо лише в разі малої розмірності конфігураційного простору в силу їх високої алгоритмічної і обчислювальної складності. У разі шестістепенного маніпулятора конфігураційний простір матиме 6 ступенів свободи, що занадто багато для його повного моделювання і використання таких алгоритмів. Протилежність комбінаторному підходу - sample-based підхід. Ідея sample-based алгоритмів полягає в тому, щоб, замість витратного повного моделювання конфігураційного простору, досліджувати його імовірнісним шляхом: розглядати набір випадкових конфігурації, відкидаючи непотрібні і складаючи карту місцевості. На

сьогодні sample-based підхід є найбільш розвинутим і використовуваним в ряді практичних завдань і показує найкращі результати [14].

Розглянемо декілька найбільш популярних sample-based алгоритмів, що використовуються при плануванні траєкторії руху.

Probabilistic Roadmap Method (PRM) – один з головних представників sample-based підходу. Основною ідеєю методу є ймовірнісна побудова карти місцевості і подальший пошук шляху з її використанням. Карта місцевості представляє з себе ненаправленої граф, вершинами якого є вільні від перетинів конфігурації робота, а ребрами - вільні шляхи між цими конфігураціями. PRM є так званим multi-query методом: після однієї відносно витратної побудови карти місцевості завдання пошуку шляху може бути вирішене з малими витратами для будь-якої кількості початкових і кінцевих положень.

Загальний алгоритм роботи PRM метода:

1. Побудова карти місцевості:

2. Генерація випадкової конфігурації робота (вершини графа) і перевірка її на перетину.

3. У разі відсутності перетинів додання вершини в карту місцевості. Інакше - перехід до п. 2.

4. Вибір кандидатів з числа існуючих вершин для з'єднання з новою вершиною.

5. З'єднання нової вершини з кандидатами і перехід на п. 2.

6. Пошук шляху по побудованій карті.

У загальному випадку конфігураційний простір заповнюється вершинами рівномірно з подальшим збільшенням щільності заповнення в "складних" областях простору. Альтернативно пропонуються різні методи заповнення конфігураційного простору: рівномірні і нерівномірні. Нерівномірний розподіл в основному направлено на збільшення щільності заповнення вершинами в областях поряд з перешкодами, або особливих складних областях. На практиці воно дає істотний приріст в швидкості лише в задачах з великими відкритими просторами і вузькими тунелями. В інших завданнях

приріст швидкості в порівнянні з рівномірним розподілом або несуттєвий, або відсутній зовсім [15][16].

Вибір кандидатів з числа наявних вершин на з'єднання з новою вершиною проводиться за принципом приєднання n найближчих вершин. В подальшому були запропоновані методи, які використовують приєднання вершини до найближчого компоненту графа, до n найближчим компонентам, або використовують принцип видимості [17].

Lazy Probabilistic Roadmap Method (Lazy PRM) – single-query метод: спрямований на знаходження шляху для однієї пари початкового і кінцевого положення. Алгоритм істотно відрізняється від оригінального PRM. Ідея полягає в тому, щоб здійснювати перевірки на перетинання тільки потенційно корисних вершин - тих, що можуть використовуватися в кінцевій траєкторії [18].

Алгоритм Lazy RPM:

1. Рівномірне заповнення конфігураційного простору випадковими вершинами і ребрами без перевірок на перетину.

2. Пошук шляху за отриманою картою місцевості.

3. Перевірка вершин і ребер шляху на перетину.

4. У разі наявності в знайденому в п. 2 шляху вершин або ребер з перетинами, видалити їх з карти місцевості і повернутися на п. 2. Інакше - шуканий шлях знайдений.

Як і в PRM, простір рівномірно заповнюється вершинами і ребрами, але без перевірок на перетину. Далі використовується локальний планувальник для знаходження оптимального шляху від початкового положення до кінцевого, і вже після цього алгоритм послідовно перевіряє вершини і ребра знайденого шляху на перетину. Якщо перетин виявлено, вершина / ребро виключається з графа, шукається новий оптимальний шлях, знову відбувається послідовна перевірка на перетин, і процес повторюється до тих пір, поки не буде знайдено шлях без перетинів.

Алгоритм дозволяє істотно підвищити швидкість знаходження траєкторії за рахунок виключення більше 99% непотрібних перевірок на перетин, які проводилися б

в оригінальному PRM. Однак, збільшується кількість операцій пошуку по графу. Особливо доцільним застосування даного методу буде в разі, коли перевірки на перетинання є витратними (складні геометричні форми об'єктів).

SPArse Roadmap Spanner algorithm (SPARS) – multi-query алгоритм. Основне завдання методу полягає в побудові оптимальної карти місцевості з мінімальним числом вершин. Це перший метод, який гарантує знаходження близько-оптимальної траєкторії з використанням кінцевих карт місцевості [19]. Алгоритми, розроблені раніше, надавали гарантії оптимальності траєкторії лише при прямуванні числа вершин карти до нескінченності. В даному ж алгоритмі ймовірність додавання нової вершини в карту місцевості прямує до нуля при збільшенні часу виконання. Метод пропонує вирішення проблеми побудови компактної карти місцевості для знаходження мінімальних шляхів у просторі.

Rapidly-exploring Random Trees (RRT) – Метод базується на ідеї послідовного дослідження конфігураційного простору шляхом побудови деревоподібного графа, що має старт у відомій початковій конфігурації. Простір досліджується шляхом приєднання до гілок дерева нових вершин в напрямках, визначених максимальною площею діаграми Вороного. Таким чином, генерація нових вершин відбувається в ще не досліджених областях.

Алгоритм генерації карти RRT:

1. Генерація нової випадкової вільної вершини.

2. Знаходження найближчій до неї існуючої вершини і спроба їх з’єднання.

Для того, щоб така структура досліджувала простір в сторону цільового положення, кожні n ітерацій (наприклад, 100) можна вибирати напрямок експансії в відповідну сторону. Даний метод дозволяє отримати гарантовано пов'язаний граф з малим числом вершин.

RRT-Connect – алгоритм є різновидом RRT. У деяких випадках пошук шляху з кінця може бути в рази ефективніше, наприклад, для перешкод, схожих на звужується шийку. У процесі виконання даного алгоритму будується не одне дерево, а два - з початковою і кінцевою конфігурації. Ці два дерева досліджують простір назустріч один одному, що підвищує ефективність роботи методу [20].

Алгоритм можна описати таким чином:

1. Приєднати до першого дереву нову вершину.

2. Спробувати поєднати цю вершину з найближчої вершиною другого дерева шляхом послідовного зростання у відповідному напрямку.

3. У разі невдачі виконати п.1-3 для другого дерева.

4. Алгоритм в цілому показує кращу швидкість виконання в порівнянні зі стандартним RRT.

Розділ 2. Кінематичний аналіз. Постановка та методика вирішення прямої та зворотної задач кінематики

2.2. Кінематика поворотів сервоприводів та переходів від однієї СК до іншої.

У якості початкової (інерціальної) системи координат було обрано СК пов’язану з першим сервоприводом. На подальших рисунках представлені кінематики поворотів (рисунок 2.1-2.6) кожного сервопривода, а також перехід від СК одного сервопривода до СК наступного (рисунок 2.7-2.10).

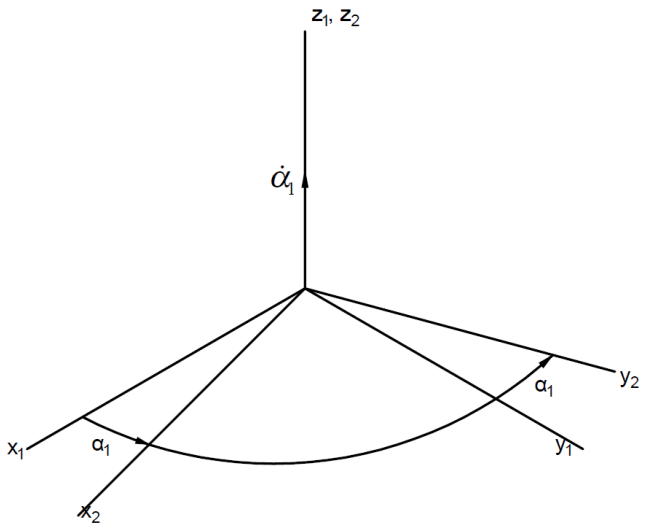

Рисунок 2.1 – Кінематика оберту першого сервопривода.

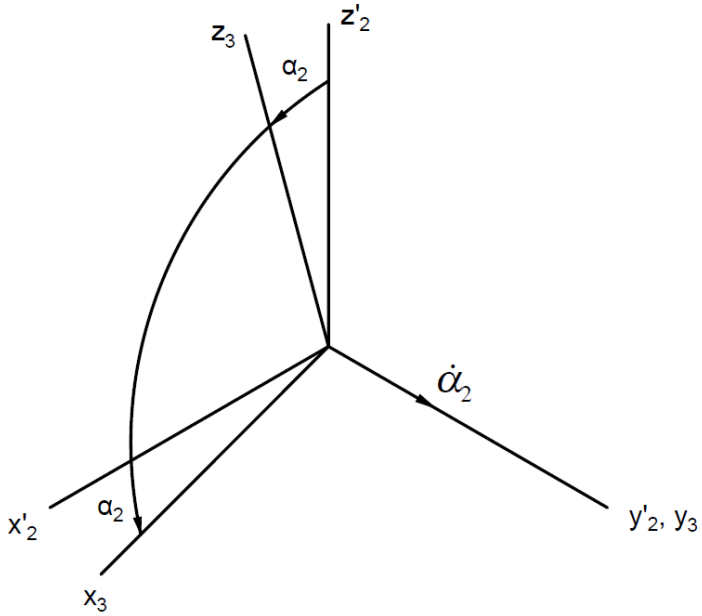

Рисунок 2.2 – Кінематика оберту другого сервопривода.

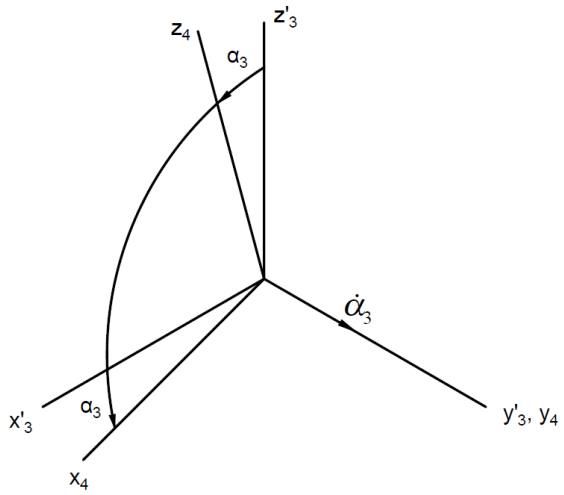

Рисунок 2.3 – Кінематика оберту третього сервопривода.

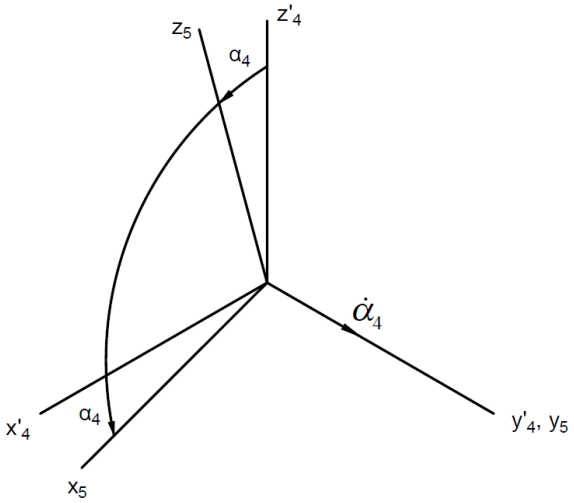

Рисунок 2.4 – Кінематика оберту четвертого сервопривода.

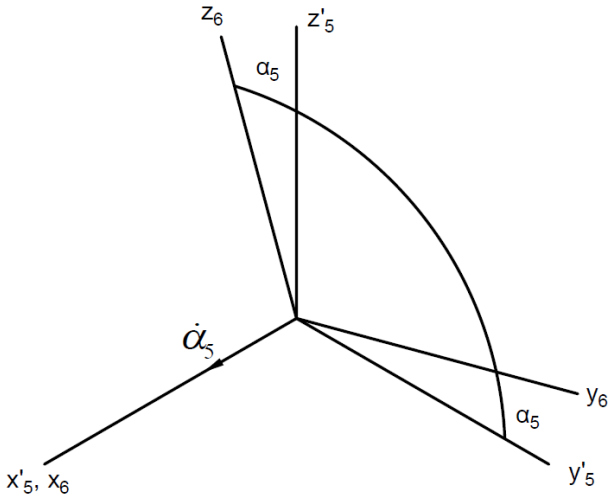

Рисунок 2.5 – Кінематика оберту п’ятого сервопривода.

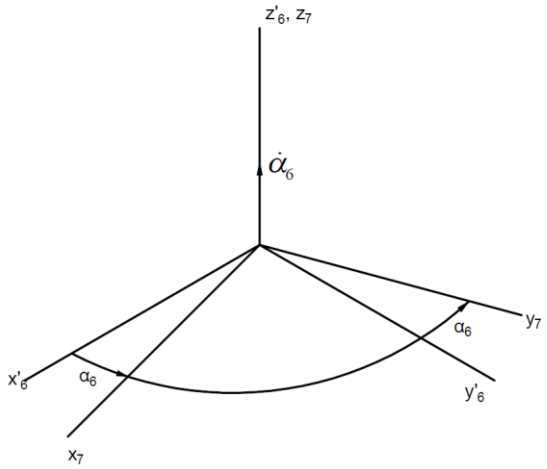

Рисунок 2.6 – Кінематика оберту шостого сервопривода.

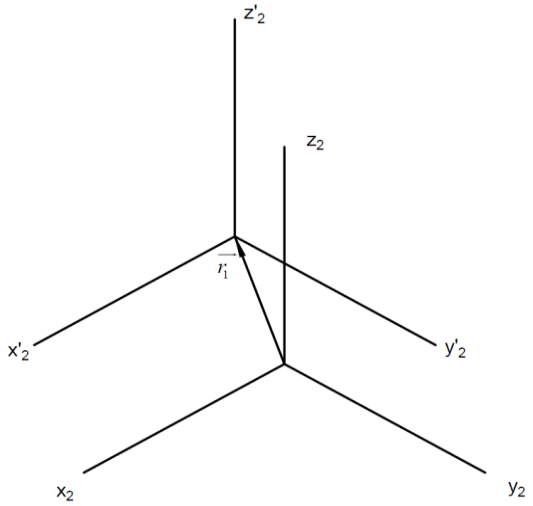

Рисунок 2.7 – перехід від СК зв’язоної з першим серворпиводом до СК другого.

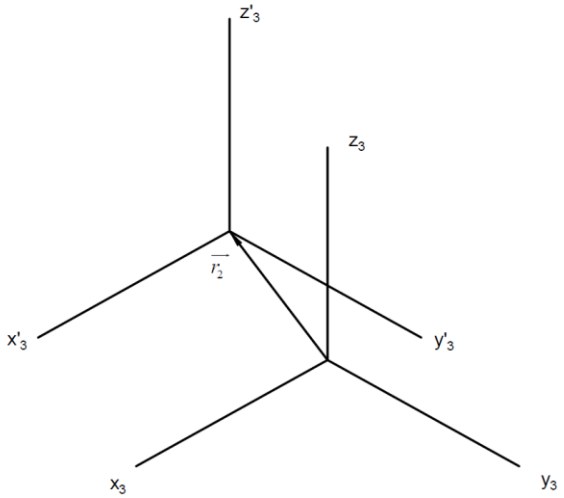

Рисунок 2.8 – перехід від СК другого серворпиводом до СК третього.

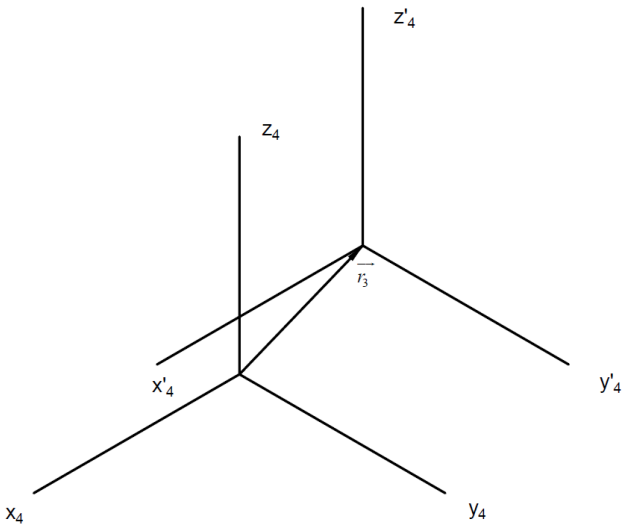

Рисунок 2.9 – перехід від СК третього серворпиводом до СК четвертого.

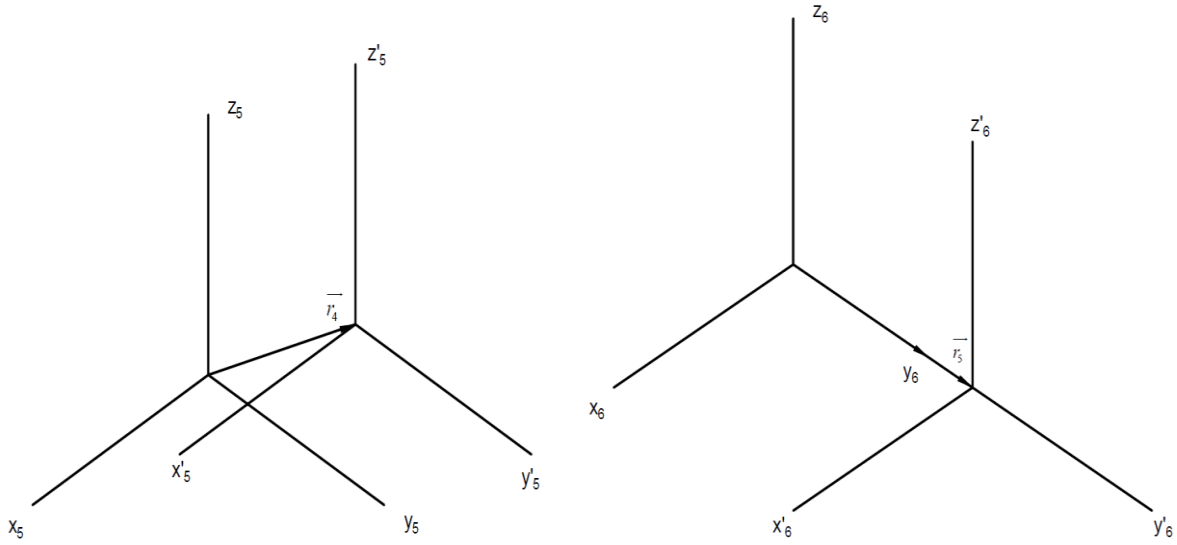

Рисунок 2.10 – а) перехід від СК четвертого серворпиводом до СК п’ятого; б) перехід від СК четвертого серворпиводом до СК п’ятого.

Після аналізу отриманих кінематик поворотів та переходів постає задача об’єднання усього в одну методику розрахунку положення захвату відносно інерціальної системи координат. Після дослідження наукової області пов’язаної з робототехнікою було прийнято рішення використати методику Денавіта-Хартенберга, алгоритм якої представлений в роботі [], з метою описання геометрії та створення кінематичної схеми.

2.2. Пряма задача кінематики

2.2.1. Алгоритм привласнення систем координат.

Очевидно, що кожне $i-e$ зчленування маніпулятора з'єднує дві ланки $i-1$ та i. Таким чином, маніпулятор з шістьма ступенями свободи має сім ланок, пронумерованих від нуля до шести, де нульове ланка відповідає «землі». Оскільки остання не відноситься безпосередньо до конструкції самого робота, незважаючи на формальне наявність семи ланок, таких роботів в літературі називають шестиланковими.

Для розуміння логіки роботи з системами координат, прив'язаними до ланок робота, підкреслимо, що $i-ma$ система координат жорстко зв'язується з $i-ою$ ланкою. Коли $i-ma$ перша ланка приводиться в рух за рахунок $i-го$ зчленування, система координат i і змінює своє положення щодо попередньої системи $i-1$.

Розглянемо алгоритм на прикладі прив'язки $i-ої$ системи координат до $i-ої$ ланки. Зауважимо, що він застосовний для всіх ланок, крім останньої. Кінцева система координат обирається особливим чином, що буде описано нижче. Отже, алгоритм складається з наступних чотирьох кроків.

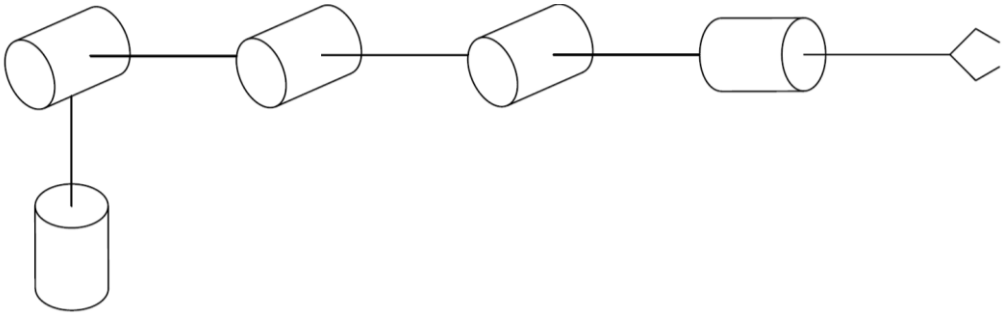

Рисунок 2.11 – Кінематична схема маніпулятора

Вибір осей z_i

Виберемо вісь z_i так, щоб вона збіглася з віссю обертання або поступального руху подальшого зчленування $i+1$ в залежності від його типу. Це означає, що відносне розташування суміжних ланок (систем координат) буде визначатися саме змінної навколо (або вздовж) цієї осі.

Вибір осей x_i

Оберемо осі x_i, $i = \{1, 2\ldots, n-1\}$ таким чином, що виконувалися дві наступні умови

Умова 1: Вісь x_i *перпендикулярна осі* z_{i-1}

Умова 2: Вісь x_i *перетинає вісь* z_{i-1}

Ось x_0 можна вибрати довільно, хоча бажано, щоб в нульової конфігурації суміжні осі x_{i-1} та x_i були співнаправлені, оскільки саме вони будуть задавати значення узагальнених координат (уздовж осей z_i), які в початковій конфігурації передбачаються нульовими. В протилежному випадку необхідно буде здавати початковий поворот, щоб нівелювати до нуля.

Точка відліку систем координат визначається перетином осей z_i і x_i. У випадках, коли x_i має безліч варіантів розташування (наприклад, якщо z_{i-1} і z_i паралельні), то, відповідно, і початок координат може розташовуватися в будь-якій точці, що належить осі z_i.

Вибір осей y_i

Виберемо вісь y_i так, щоб система координат, задана одиничними векторами $\vec{x}_i$, $\vec{y}_i$, $\vec{z}_i$ була правою, тобто в напрямку, заданому векторним добутком:

$$\vec{y}_i = \vec{z}_i \times \vec{x}_i, \qquad (2.1)$$

Вибір системи координат n

Завершальним кроком, виберемо систему координат n, пов'язану зі схватом або робочим інструментом. Для більшості сучасних роботів останнє зчленування є обертальним, через що осі z_{n-1} і z_n збігаються. В цьому випадку шукана система виходить шляхом зсуву попередньої системи координат на фіксоване значення d_n уздовж осі z_{n-1} і подальшого повороту на змінний кут θ_n навколо z_{n-1} або навпаки.

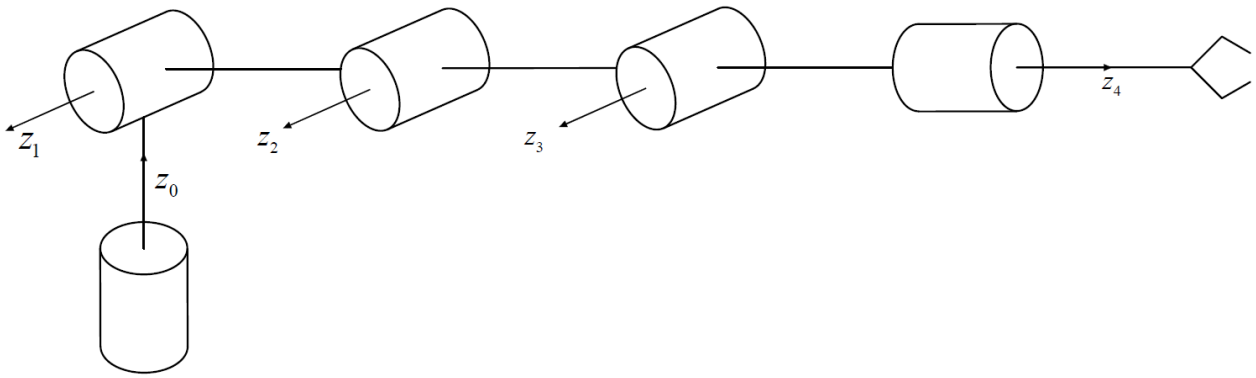

Рисунок 2.12 – Вибір осей z_i

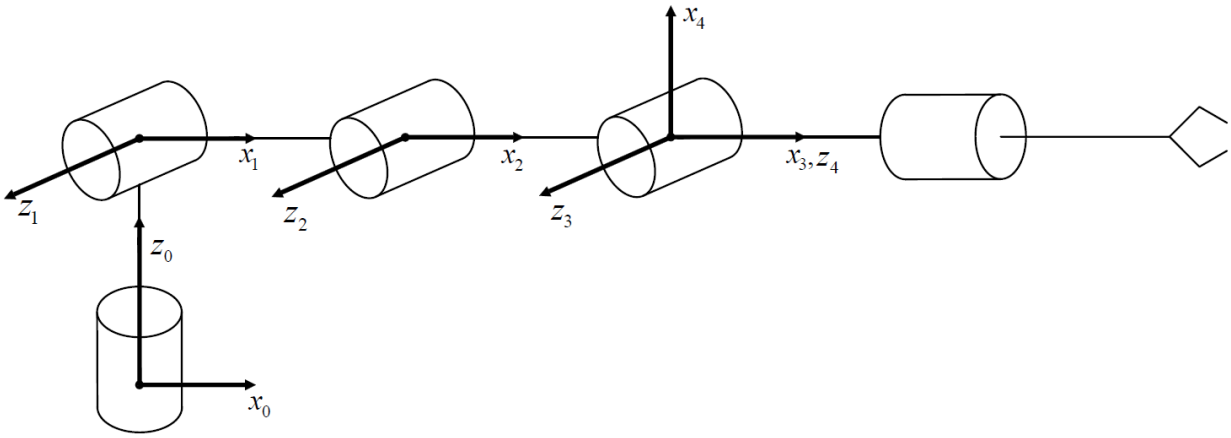

Рисунок 2.13 – Вибір осей x_i

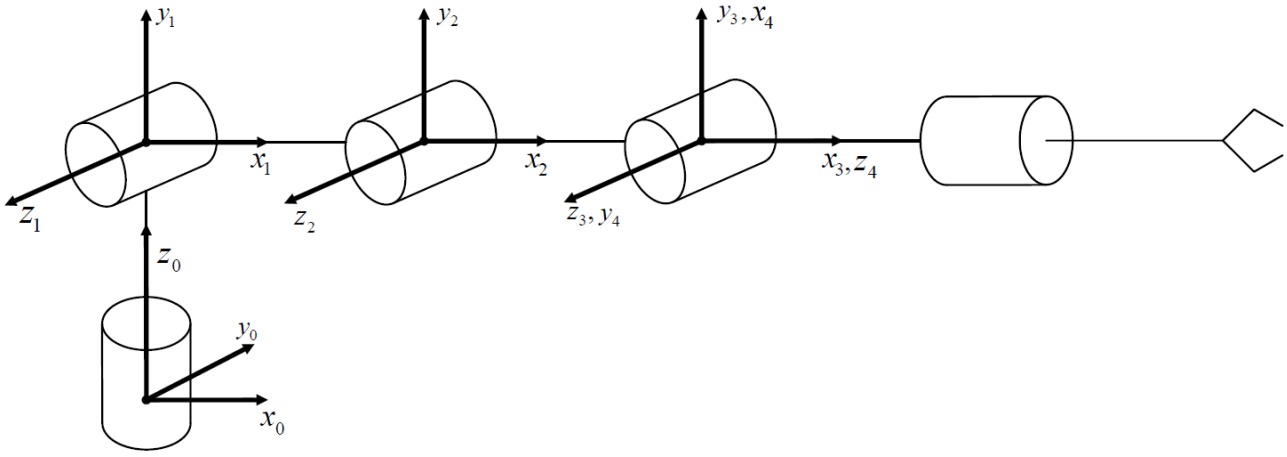

Рисунок 2.14 – Вибір осей

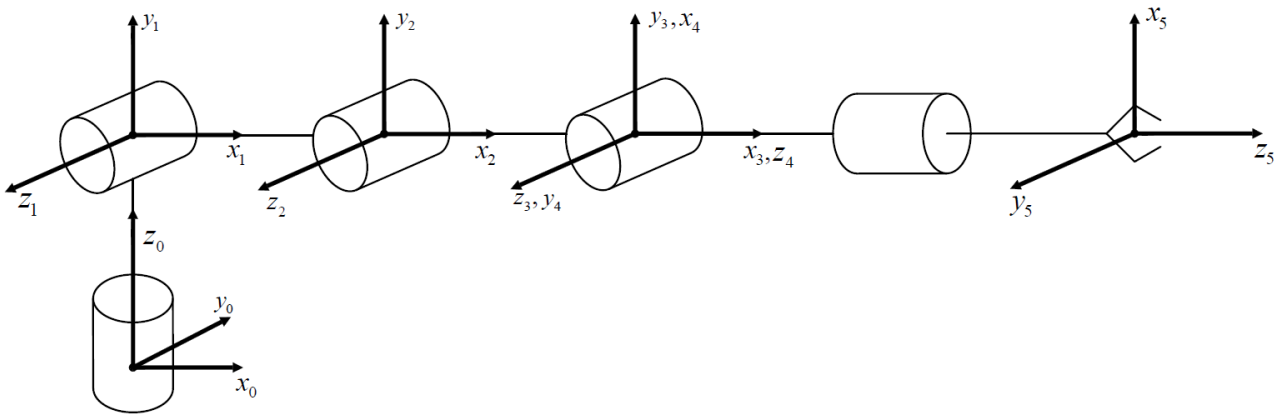

Рисунок 2.15 – Вибір системи координат n

2.2.1. Визначення параметрів Денавіта-Хартенберга

метод Денавіта-Хартенберга дозволяє зменшити кількість координат, які однозначно визначають тіло (систему координат) в просторі, з шести до чотирьох, відомих як параметри Денавіта-Хартеберга:

- a_i – відстань вздовж осі x_i від z_{i-1} до z_i;
- α_i – кут навколо осі x_i від z_{i-1} до z_i;
- d_i – відстань вздовж осі z_{i-1} від x_{i-1} до x_i;
- θ_i – кут навколо осі z_{i-1} від x_{i-1} до x_i;

Параметри a_i і α_i завжди є константами для всіх кінематичних схем і обумовлені конструкцією маніпуляторів. Що стосується решти параметрів d_i та θ_i, серед них тільки один параметр є постійним, а інший - змінним залежно від типу зчленування: в разі обертального - кут θ_i змінний, зміщення d_i постійне, в разі поступального - навпаки.

Таблиця 2.1 – Параметри Денавіта-Хартенберга

Ланка, i	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	d_1	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3
4	0	$\pi/2$	0	$\theta_4+\pi/2$
5	0	0	d_5	θ_5

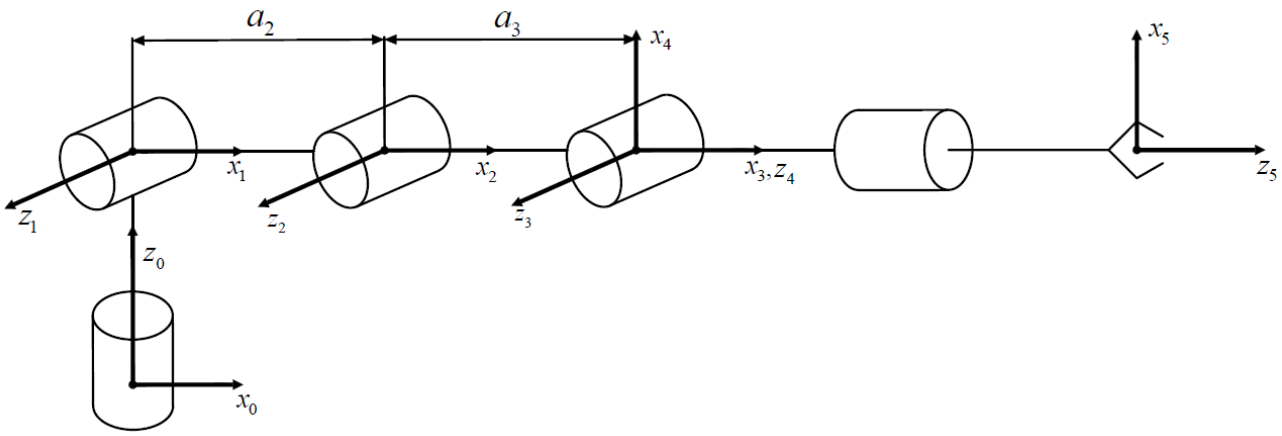

Рисунок 2.16 – Визначення параметрів a_i

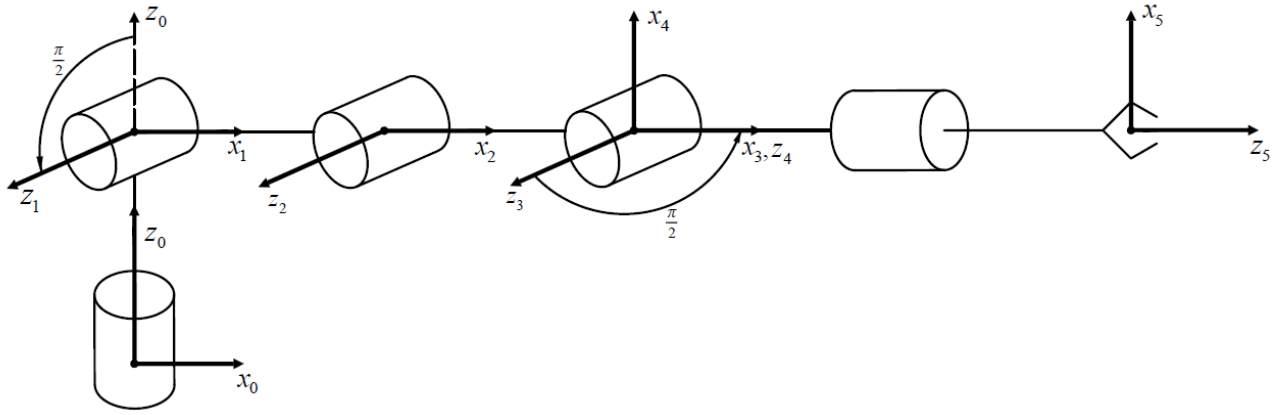

Рисунок 2.17 – Визначення параметрів α_i

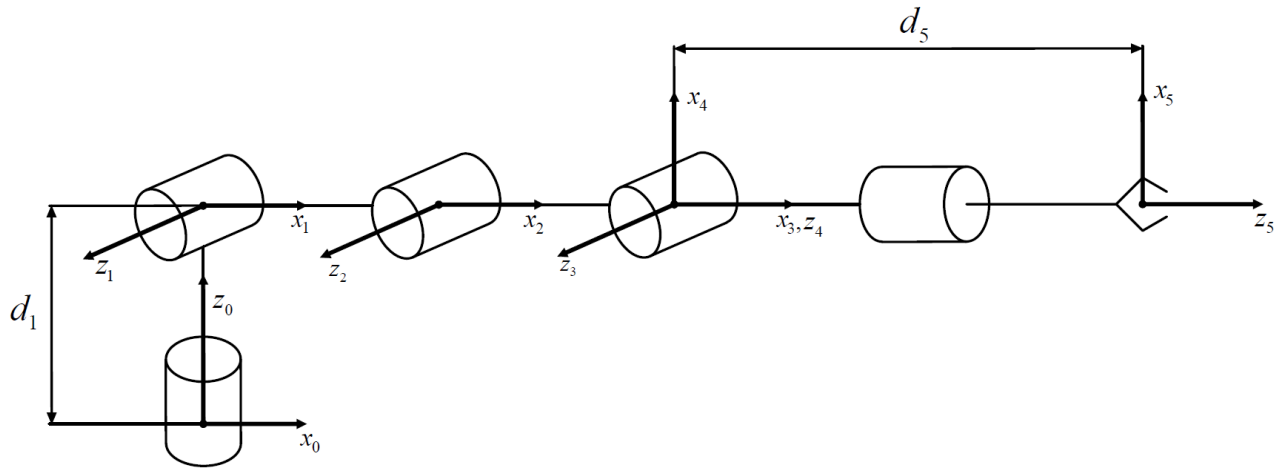

Рисунок 2.18 – Визначення параметрів d_i

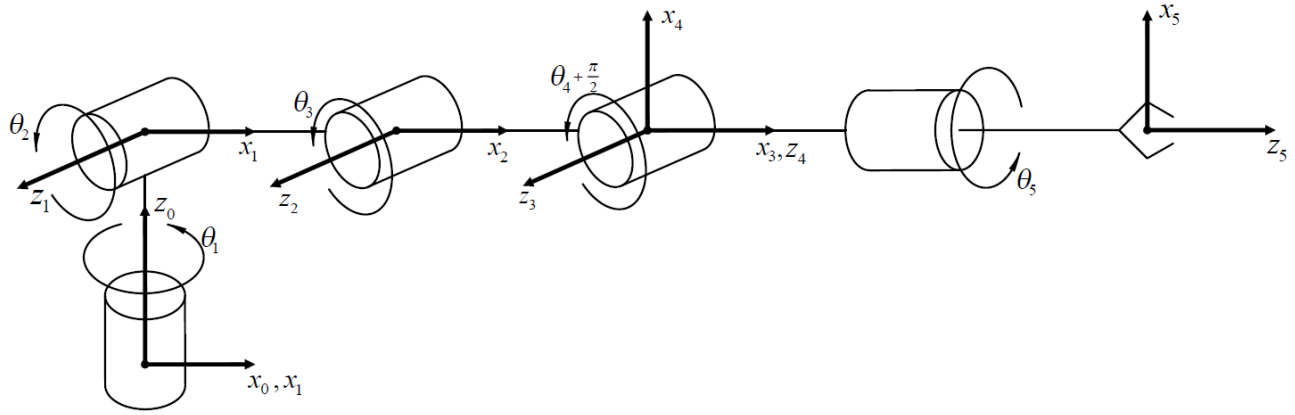

Рисунок 2.19 – Визначення параметрів θ_i

З рисунка 2.16 помітно, що в даному випадку маємо два ненульових параметри a_2 та a_3, котрі відповідають довжинам відповідних ланок. На рисунку 7 позначені ненульові кути α_1 і α_4. Слід відмітити, що при визначенні цих кутів важливо враховувати знаки які залежать від напрямку обороту, в даному випадку усі кути являється додатними, оскільки оберт здійснюється проти годинникової стрілки.

Лінійні зміщення d_1 та d_5 позначені на рисунку 8. Оскільки усі зчленування маніпулятора обертальні, то відповідно і ці параметри являється постійними.

Кути $\theta_1 - \theta_6$ являються узагальненими координатами, тобто змінними величинами, представлені на рисунку 9.

Отримані параметри Денавіта-Хартенберга представлені в Таблиці 1.

2.2.2. Матричні однорідні перетворення

Пряма задача кінематики полягає в обчисленні координат системи, пов'язаної зі схватом або робочим інструментом, в залежності від узагальнених координат маніпулятора.

Зрозуміло, що при розв’язанні ПЗК розглядаються дві системи координат: вихідна (інерційна) $o_0x_0y_0z_0$ та кінцева, пов’язана зі схватом $o_nx_ny_nz_n$. Відомо, що розташування цих двох систем одна відносно одної визначається трьома лінійними та трьома кутовими координатам.

Якщо розглядати два набори координат k^0 та k^n однієї і тієї ж точки у просторі, що виражені відносно систем $o_0x_0y_0z_0$ і $o_nx_ny_nz_n$ отримаємо наступне:

$$k^0 = T_n^0 k^n, \tag{2.2}$$

де T_n^0 – перетворення, що несе інформацію про лінійні зміщення і просторову орієнтацію однієї системи відносно іншої.

Матриця T_n^0, шо визначає зв'язок систем координат $o_0x_0y_0z_0$ і $o_nx_ny_nz_n$ має назву матриця однорідних перетворень і виглядає:

$$T_n^0 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_n^0 & s_n^0 & a_n^0 & p_n^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_n^0 & p_n^0 \\ 0 & 1 \end{bmatrix}, \tag{2.3}$$

де вектори n_n^0, s_n^0 та a_n^0 виражають направлення осей x_n, y_n і z_n відносно системи координат $o_0x_0y_0z_0$, R_n^0 – матриця обертання системи координат $o_nx_ny_nz_n$ відносно $o_0x_0y_0z_0$, p_n^0 - вектор лінійних зміщень початку координат системи $o_nx_ny_nz_n$ відносно $o_0x_0y_0z_0$.

Як можливо помітити матриця T_n^0 має розмірність (4×4), тому виникає необхідність розширити вектори координат k^0 та k^n одиницею для відповідності розмірностей:

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = T_n^0 \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix} \tag{2.4}$$

Перерахуємо найважливіші властивості та характеристики котрі будуть застосовуватися:

1. Оберт на нульовий кут визначається одиничною матрицею:

$$R_{\alpha=0}=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}=I, \tag{5}$$

2. Оберт у від'ємному напрямку визначається:

$$R_{-\alpha}=R_{\alpha}^{-1}=R_{\alpha}^{T}, \tag{2.6}$$

3. Існують бузові матриці обертання навколо осей x, y та z:

$$R_{x,\alpha}=\begin{bmatrix}1&0&0\\0&\cos\alpha&-\sin\alpha\\0&\sin\alpha&\cos\alpha\end{bmatrix}, \tag{2.7}$$

$$R_{y,\alpha}=\begin{bmatrix}\cos\alpha&0&\sin\alpha\\0&1&0\\-\sin\alpha&0&\cos\alpha\end{bmatrix}, \tag{2.8}$$

$$R_{z,\alpha}=\begin{bmatrix}\cos\alpha&-\sin\alpha&0\\\sin\alpha&\cos\alpha&0\\0&0&1\end{bmatrix}, \tag{2.9}$$

4. Послідовність обертів навколо поточних осей визначається домноженням з права. Наприклад перетворення параметризоване кутами Ейлера φ, θ, ψ:

$$R_{zyz}=R_{z,\varphi}R_{y,\theta}R_{z,\psi}=$$

$$=\begin{bmatrix}C_\varphi&-S_\varphi&0\\S_\varphi&C_\varphi&0\\0&0&1\end{bmatrix}\begin{bmatrix}C_\theta&0&S_\theta\\0&1&0\\-S_\theta&0&C_\theta\end{bmatrix}\begin{bmatrix}C_\psi&-S_\psi&0\\S_\psi&C_\psi&0\\0&0&1\end{bmatrix}=$$

$$=\begin{bmatrix}-S_\varphi S_\psi+C_\varphi C_\theta C_\psi&-C_\psi S_\varphi-C_\varphi C_\theta S_\psi&C_\varphi S_\theta\\C_\varphi S_\psi+C_\theta C_\psi S_\varphi&C_\varphi C_\psi-C_\theta S_\varphi S_\psi&S_\varphi S_\theta\\-C_\psi S_\theta&S_\theta S_\psi&C_\theta\end{bmatrix}, \tag{2.10}$$

де $C_\alpha=\cos\alpha$, $S_\alpha=\sin\alpha$, $\alpha=\{\varphi,\theta,\psi\}$.

На початку були визначені чотири параметри для кожної ланки маніпулятора. Тепер з них необхідно побудувати відповідні матриці однорідного перетворення наступним чином:

$$T_i = T_{z,\theta_i} T_{z,d_i} T_{x,a_i} T_{x,\alpha_i} =$$

$$= \begin{bmatrix} R_{z,\theta_i} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & p_{d_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & p_{a_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{x,\alpha_i} & 0 \\ 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (2.11)$$

де i – номер ланки; R_{z,θ_i}, R_{x,α_i} – базові матриці обертання; p_{d_i} та p_{a_i} – вектори з ненульовими компонентами $p_z = d_i$ і $p_x = a_i$, з чого слідує:

$$p_{d_i} = \begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix}, \; p_{a_i} = \begin{bmatrix} a_i \\ 0 \\ 0 \end{bmatrix}, \qquad (2.12)$$

Підставляючи параметри Денавіта-Хартенберга, отримаємо n матриць однорідного перетворення:

$$T_1^0 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (2.13)$$

$$T_2^1 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (2.14)$$

$$T_3^2 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & a_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.15}$$

$$T_4^3 = \begin{bmatrix} \cos(\theta_4+\pi/2) & 0 & \sin(\theta_4+\pi/2) & 0 \\ \sin(\theta_4+\pi/2) & 0 & -\cos(\theta_4+\pi/2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.16}$$

$$T_5^4 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.17}$$

Загальну матрицю, котра зв'язує усі системи координат можливо отримати послідовним множенням:

$$T_n^0 = T_1^0 T_2^1 \dots T_n^{n-1} = \begin{bmatrix} R_n^0 & p_n^0 \\ 0 & 1 \end{bmatrix} \tag{2.18}$$

де матриця обертання R_n^0 та вектор p_n^0 задають, орієнтацію та положення системи координат, що зв'язана із захватом, відносно початкової системи.

2.2.3. Розрахунок кутів Ейлера

Мета прямої задачі кінематики – знаходження лінійних та кутових координат системи $o_n x_n y_n z_n$, пов'язаної із захватом, відносно базової системи $o_0 x_0 y_0 z_0$. Попередньо була отримана матриця однорідного перетворення T_n^0, котра містить в собі необхідну інформацію, представлену у вигляді матриці обертів R_n^0 та вектору p_n^0.

Використання матриці обертання R_n^0 являється не самим оптимальним рішенням, оскільки по її вигляду іноді складно визначити орієнтацію яку вона задає. Саме тому постає задача отримання кутових координат у явному вигляді.

Проведемо параметризацію матриці обертання використавши кути Ейлера.

Матриця обертів, що задається кутами Ейлера має вигляд:

$$R_n^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} =$$

$$= \begin{bmatrix} -S_\varphi S_\psi + C_\varphi C_\theta C_\psi & -C_\psi S_\varphi - C_\varphi C_\theta S_\psi & C_\varphi S_\theta \\ C_\varphi S_\psi + C_\theta C_\psi S_\varphi & C_\varphi C_\psi - C_\theta S_\varphi S_\psi & S_\varphi S_\theta \\ -C_\psi S_\theta & S_\theta S_\psi & C_\theta \end{bmatrix}, \qquad (2.19)$$

Суть завдання полягає у визначенні кутів Ейлера φ, θ, ψ маючи матрицю обертання R_n^0. З метою вирішення даної задачі необхідно розглянути декілька випадків, що залежать від значення елемента r_{33}.

1. Випадок коли $r_{33} \neq \pm 1$

При такому випадку $\sin\theta \neq 0$. Застосуємо основну тригонометричну рівність:

$$\sin^2\theta + \cos^2\theta = 1, \qquad (2.20)$$

$$\sin\theta = \pm\sqrt{1-\cos^2\theta} = \pm\sqrt{1-r_{33}}, \qquad (2.21)$$

звідки випливає, що кут θ можливо отримати за наступної рівності:

$$\theta = \text{atan2}\left(\pm\sqrt{1-r_{33}}, r_{33}\right). \qquad (2.22)$$

В залежності від обраного знаку перед коренем у (22) змінюються вираз для розрахунку двох інших кутів ϕ та ψ:

$$\varphi = \text{atan2}\left(\pm r_{23}, \pm r_{13}\right), \qquad (2.23)$$

$$\psi = \text{atan2}\left(\pm r_{32}, \mp r_{31}\right). \qquad (2.24)$$

2. Випадок коли $r_{33} = 1$

При такій умові $\cos\theta = 1$, $\sin\theta = 0$, звідки випливає $\theta = 0$, в результаті чого матриця R_n^0 приймає вигляд:

$$R_n^0 = \begin{bmatrix} -S_\varphi S_\psi + C_\varphi C_\psi & -C_\psi S_\varphi - C_\varphi S_\psi & 0 \\ C_\varphi S_\psi + C_\psi S_\varphi & C_\varphi C_\psi - S_\varphi S_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} C_{\varphi+\psi} & -S_{\varphi+\psi} & 0 \\ S_{\varphi+\psi} & C_{\varphi+\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (2.25)$$

При такому випадку отримуємо невизначеність рішення, у зв'язку з тим, що можливо визначити тільки суму кутів $\varphi+\psi$:

$$\varphi+\psi = \operatorname{atan2}\left(r_{21}, r_{11}\right). \qquad (2.26)$$

3. Випадок коли $r_{33} = -1$

При такій умові $\cos\theta = -1,\ \sin\theta = 0$, звідки випливає $\theta = \pi$, в результаті чого матриця R_n^0 приймає вигляд:

$$R_n^0 = \begin{bmatrix} -S_\varphi S_\psi - C_\varphi C_\psi & -C_\psi S_\varphi + C_\varphi S_\psi & 0 \\ C_\varphi S_\psi - C_\psi S_\varphi & C_\varphi C_\psi + S_\varphi S_\psi & 0 \\ 0 & 0 & -1 \end{bmatrix} =$$

$$= \begin{bmatrix} -C_{\varphi-\psi} & -S_{\varphi-\psi} & 0 \\ S_{\varphi-\psi} & C_{\varphi-\psi} & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & -1 \end{bmatrix}. \qquad (2.27)$$

Такий випадок також веде до невизначеності рішення, оскільки можливо знайти тільки різницю кутів $\varphi-\psi$:

$$\varphi-\psi = \operatorname{atan2}\left(r_{21}, r_{22}\right). \qquad (2.28)$$

З розглянутих випадків помітно, що при $r_{33} = \pm 1$ виникає неоднозначність визначення кутів Ейлера. Тому в таких випадках пропонується прийняти один з кутів рівним 0 та обчислити інший.

2.3. Зворотна задача кінематики

Сутність ЗЗК полягає у розрахунку узагальнених координат при заданих лінійних та кутових координатах захвату маніпулятора. Дана задача являється дещо складнішою ніж ПЗК, оскільки може привести до невизначеності рішення, це означає, що одному набору положення захвату у просторі, можуть відповідати різні конфігурацію робота. Також варто зазначити, що рішення ЗЗК суттєво залежить від конструкції маніпулятора, у зв'язку з чим виключається можливість створення єдиного способу рішення ЗЗК у загальному випадку.

При рішенні ЗЗК вихідними даними являються:

- Три кутові координати;
- Три лінійні координати;
- Параметри Денавіта-Хартенберга, котрі залежать від конструкції маніпулятора.

Саму по собі зворотну задачу кінематики розділяють на дві підзадачі:

- ЗЗК по положенню
- ЗЗК по орієнтації

2.3.1. Зворотна задача кінематики по положенню

Вирішення задачі проводиться від положення "сферичне зап'ястя" – це таке конструктивне розташування останніх трьох обертових зчленувань маніпулятора, при якому їх осі обертання перетинаються в одній точці. У нашому випадку осі перетинаються в точці o_3.

Застосувавши суму векторів (рисунок 2.10)

$$p_5^0 = p_3^0 + d_5 R_5^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad (2.29)$$

виразимо координати точки o_3:

$$p_3^0 = p_5^0 - d_5 R_5^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad (2.30)$$

де $p_3^0 = \begin{bmatrix} x_3^0 & y_3^0 & z_3^0 \end{bmatrix}^T$.

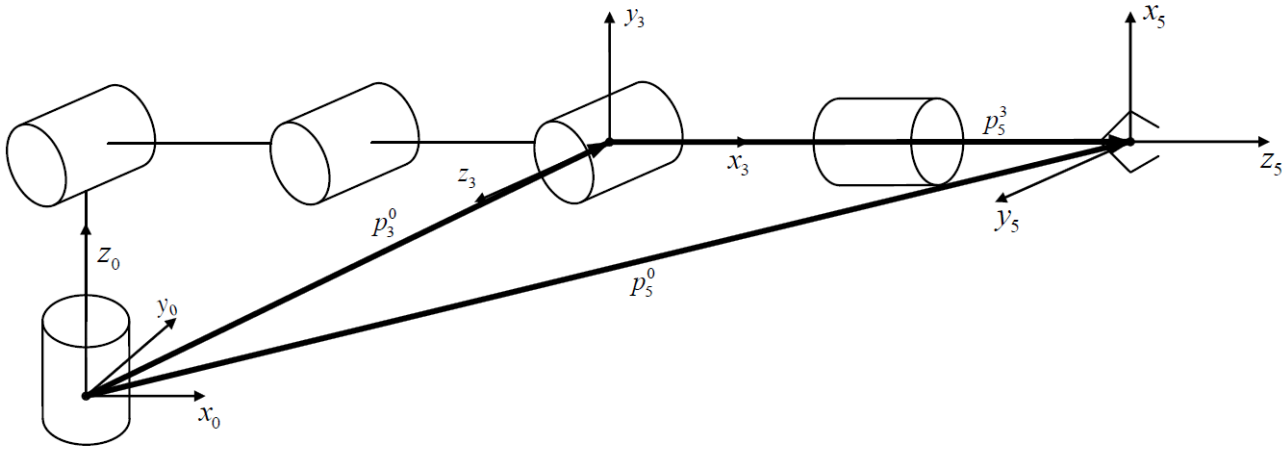

Рисунок 2.20 – Сума векторів при кінематичній декомпозиції

Враховуючи положення в якому знаходиться маніпулятор знайдемо першу узагальнену координату:

$$\theta_1 = \text{atan2}\left(y_3^0, x_3^0\right). \qquad (2.31)$$

Не складно помітити, що вже на даному етапі з'являються невизначеності рішення ЗЗК, оскільки одному і тому ж значення тангенса відповідають два значення координати, що відрізняються на кут π, тому, альтернативно, координату можливо розрахувати наступним чином:

$$\theta_1 = \text{atan2}\left(y_3^0, x_3^0\right) + \pi. \qquad (2.32)$$

Варто відзначити, що від вибору виразу (2.31) або (2.32) залежить вигляд співвідношень для θ_2 та θ_3. В даному випадку перевагу надано використанню рішення (2.31).

Слід зазначити, що вирази (2.31) та (2.32) мають рішення у випадках коли $x_3^0 \neq 0$ і $y_3^0 \neq 0$, це випадки коли напрям вектору p_3^0 не співпадає з напрямом осі z_0. Тобто

використання представлених вище співвідношень можливе якщо точка перетину сферичного зап'ястя не лежить на осі z_0, в інакшому випадку отримуємо ситуацію коли узагальнена координата θ_1 має безліч значень.

Для подальшого розрахунку узагальнених координат введемо три відрізки довжини який запишемо наступним чином:

$$a = \sqrt{\left(x_3^1\right)^2 + \left(y_3^1\right)^2 + \left(z_3^1\right)^2}, \tag{2.33}$$

$$b = \left(z_3^0 - d_1\right), \tag{2.34}$$

$$c = \sqrt{\left(x_3^0\right)^2 + \left(y_3^0\right)^2}. \tag{2.35}$$

Дані три відрізка між собою утворюють прямокутний трикутник, де c – проекція відстань від o_1 до o_3 на площину x_0y_0, b – зміщення по висоті o_3 відносно o_1, a – являється гіпотенузою в трикутнику.

Використавши теорему Піфагора, запишемо:

$$a^2 = b^2 + c^2. \tag{2.36}$$

Застосувавши теорему косинусів та формулу різниці кутів, отримаємо:

$$a^2 = a_2^2 + a_3^2 - 2a_2a_3\cos\left(\pi - \theta_3\right) = a_2^2 + a_3^2 + 2a_2a_3\cos\theta_3. \tag{2.37}$$

Об'єднавши вирази (37) та (38) отримаємо:

$$b^2 + c^2 = a_2^2 + a_3^2 + 2a_2a_3\cos\theta_3, \tag{2.38}$$

звідки можливо виразити $\cos\theta_3$:

$$\cos\theta_3 = \frac{b^2 + c^2 - a_2^2 - a_3^2}{2a_2a_3}. \tag{2.39}$$

Застосовуючи основну тригонометричну тотожність виразимо $\sin\theta_3$:

$$\sin\theta_3 = \pm\sqrt{1 - \cos^2\theta_3}. \tag{2.40}$$

В результаті, узагальнену координату θ_3 можливо обчислити за наступним виразом:

$$\theta_3 = \text{atan2}\left(\pm\sqrt{1-\cos^2\theta_3}, \cos\theta_3\right), \tag{2.41}$$

З ціллю визначення кута θ_2 розглянемо різницю двох кутів:

- Кута α, котрий створений відрізками a та c;
- Кута β, котрий створений відрізками a та a_2.

Узагальнену координату θ_2 виразимо наступним чином:

$$\theta_2 = \alpha - \beta. \tag{2.42}$$

Враховуючи тригонометричні вирази для тангенсів кутів α та β

$$\tan\alpha = \frac{b}{c}, \tag{2.43}$$

$$\tan\beta = \frac{a_3 \sin\theta_3}{a_2 + a_3\cos\theta_3}, \tag{2.44}$$

перепишемо вираз (42) у вигляді:

$$\theta_2 = \text{atan2}(b,c) - \text{atan2}\left(a_3\sin\theta_3, a_2 + a_3\cos\theta_3\right). \tag{2.45}$$

В кінцевому підсумку, були отримані вирази для перших трьох узагальнених координат q_1, q_2 та q_3.

2.3.2. Зворотна задача кінематики по орієнтації

Відомо, що комбінація послідовних обертів навколо однієї осі визначається перемноженням з правою стороні, тобто матрицю R_6^0 можливо представити наступним чином:

$$R_6^0 = R_3^0 R_6^3, \tag{2.46}$$

де R_6^0 задано по умові, а R_3^0 обчислюється на етапі рішення прямої задачі.

Виразим R_6^3 використовуючи ортогональну властивість матриці обертання:

$$R_6^3 = \left(R_3^0\right)^{-1} R_6^0 = \left(R_3^0\right)^T R_6^0. \tag{2.47}$$

Зважаючи на конструкцію сферичного зап'ястя останні три ланки забезпечують орієнтацію захвату у відповідності з матрицею R_6^3 за допомогою останніх трьох

узагальнених координат, котрі співпадають з кутами Ейлера, в результаті формуючи матрицю:

$$R_6^3 = R_{zyz} = R_{z,\varphi} R_{y,\theta} R_{z,\psi} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \qquad (2.48)$$

Таким чином, узагальнені координати які залишилися можуть бути розраховані наступним чином:

$$q_4 = \theta = \text{atan2}\left(\pm\sqrt{1 - r_{33}}, r_{33}\right), \qquad (2.49)$$

$$q_5 = \phi = \text{atan2}\left(\pm r_{23}, \pm r_{13}\right), \qquad (2.50)$$

$$q_6 = \psi = \text{atan2}\left(\pm r_{32}, \mp r_{31}\right). \qquad (2.51)$$

Слід відмітити що у випадках коли $r_{33} = \pm 1$ виникає неоднозначність визначення кутів Ейлера. На практиці досить часто такі ситуації уникаються шляхом присвоєння наближених значень.

Розділ 3. Динаміка та створення алгоритму керування

3.1. Динамічна модель

Керування зчленуванням обертового типу фактично зводиться до керування електроприводом з зубчастою передачею, динамічну модель якого можливо представити у вигляді двох складових: електричної та механічної.

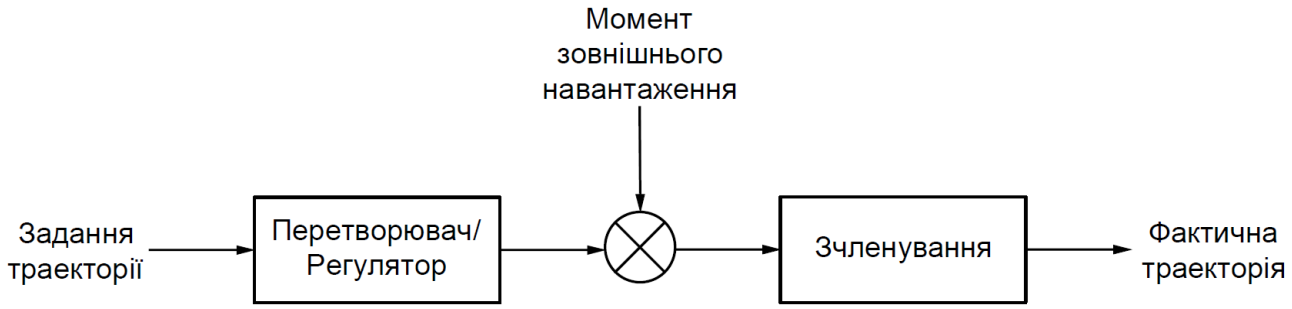

Рисунок 3.1 – Функціональна схема керування одноступеневим обертальним зчленуванням

Електрична складова, задається електричним колом котре складається з індуктивності, резистора та двигуна і визначається наступним виразом:

$$Li(t)+Ri(t)=u(t)-K_\varepsilon\omega(t)=u(t)-K_\varepsilon\dot{\theta}(t), \qquad (3.1)$$

де L, R, $i(t)$, $u(t)$ – індуктивність, опір, сила струму та напруга на якорі, K_ε – коефіцієнт проти-ЕРС, $\omega(t)$, $\theta(t)$ – кутова швидкість та кут повороту ротора.

Характер механічної складової, обумовленої зубчастою передачею, можливо записати наступним чином:

$$J\ddot{\theta}(t)+K_f\dot{\theta}(t)=K_\mu i(t)-\mu(t), \qquad (3.2)$$

де J – сумарний момент інерції зубчастої передачі та двигуна, K_f – коефіцієнт тертя, K_μ – коефіцієнт моменту сили, $\mu(t)$ – момент навантаження.

Застосувавши перетворення Лапласа до представлених вище рівнянь динаміки отримаємо:

$$(Lp+R)I(p)=U(p)-K_\varepsilon p\Theta(p), \qquad (3.4)$$

$$(Jp+K_f)p\Theta(p)=K_\mu I(p)-\mathrm{M}(p). \qquad (3.5)$$

На основі отриманих зображень запишемо передатну функцію відношення входу $U(p)$ до виходу $\Theta(p)$, приймаючи величину моменту навантаження рівну 0:

$$\frac{\Theta(p)}{U(p)} = \frac{K_\mu}{p\left((Lp+R)\left(Jp+K_f\right)+K_\varepsilon K_\mu\right)}. \qquad (3.6)$$

З отриманої залежності структурну схему можливо представити як показано на рисунку 3.2.

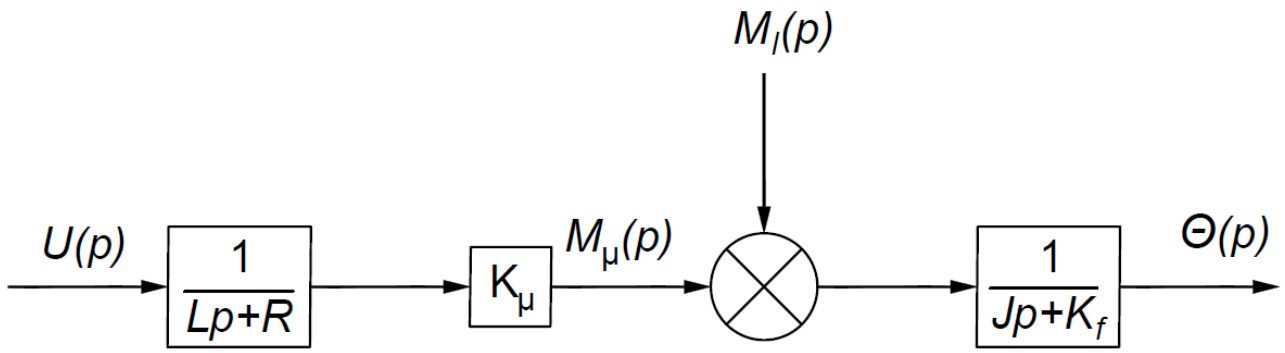

Рисунок 3.2 – Структурна схема обертального зчленування.

Слід зазначити, що схема представлена на рисунку 3.2 не являється універсальною, оскільки в залежності від будови приладу вона може змінюватися. Альтернативно, досить розповсюдженою являється структурна схема з наявністю зворотного зв’язку в ній.

Зворотній зв’язок, здебільшого, використовується у випадках коли за маніпулятором ведеться постійний нагляд, або ж, сам маніпулятор обладнаний органам відчуттів. В обох випадках зворотній зв’язок використовується з метою надання можливості корегування кінцевого положення у випадках коли воно не відповідатиме заданому (реалізація схеми зворотного зв’язку по відхиленню).

3.2. Створення алгоритму з використанням можливостей Arduino

Розробка алгоритму керування маніпулятором використовуючи середовище Arduino являється задачею непростою. Складність реалізації поставленої задачі полягає не тільки у факті того що це структурне програмування, хоча це також вносить деякі незручності, але і у тому, що в самій Arduino не існує функцій та методів для вирішення поставленої задачі.

Саме тому, перед початком виконання, було прийнято рішення провести аналіз питання і спробувати знайти методики для поліпшення складності. За результатами дослідження предметної області було виявлено, що задача керування сервоприводами за допомогою плат Arduino поставала і раніше, через що, на даний момент вже існують програмні рішення, у вигляді бібліотек, котрі дозволяють суттєво спростити реалізацію руху сервоприводів.

У якості допоміжного інструменту для реалізації алгоритму керування було обрано бібліотеку *iarduino_MultiServo.* Дана бібліотека створена з метою поліпшення процесу керування сервоприводами використовуючи Adruino. Найголовнішим плюсом у використанні даної бібліотеки є факт того, що керування відбувається за допомогою ШИМ, що в свою чергу дозволяє реалізувати можливість регулювання швидкості повороту сервоприводу.

Розглянемо декілька ключових функцій даної бібліотеки які використовувалися при розробці алгоритму:

- Функція **servoSet(**№ виходу, Параметри**)** – функція використовується для встановлення необхідних параметрів для сервоприводів. Перший вхідний параметр відповідає номеру виходу до якого підключено сервопривод, другий параметр відповідає за конфігурацію яку буде призначено.
- Функція **servoWrite((**№ виходу, кут**) –** функція призначення саме для повороту сервоприводів. У якості вхідних параметрів приймає номер виходу до якого під’єднано сервопривод, та кут до якого необхідно його повернути.
- Функція **begin() –** використовується для ініціалізації початку роботи.

Лістинг 3.1 – Вихідний код програми для реалізації переміщення об’єкта.

```
#include <iarduino_MultiServo.h>
iarduino_MultiServo MSS;

  void setup() {
    MSS.servoSet(10, SERVO_MG996R);//1
    MSS.servoSet(11, SERVO_MG996R);//2
    MSS.servoSet(12, SERVO_MG996R);//3
    MSS.servoSet(13, SERVO_MG996R);//4
    MSS.servoSet(14, SERVO_MG996R);//5
    MSS.servoSet(15, SERVO_MG996R);//6
    MSS.begin();

    //Виставлення в початкове положення

  servoSlow(15, 50, 50, 30);
  delay(500);
  servoSlow(14, 0, 0, 30);
  delay(500);
  servoSlow(13, 0, 0, 30);
  delay(500);
  servoSlow(12, 130, 130, 30);
  delay(500);
  servoSlow(11, 85, 85, 30);
  delay(500);
  servoSlow(10, 0, 0, 30);
  delay(500);


  }

  void loop() {

    servoSlow(15, 50, 0, 30);
    delay(500);
    servoSlow(14, 0, 50, 30);
    delay(500);
    servoSlow(13, 0, 0, 30);
    delay(500);
    servoSlow(12, 130, 155, 30);
    delay(500);
    servoSlow(11, 85, 70, 30);
    delay(500);
    servoSlow(10, 0, 105, 30);
    delay(500);
    servoSlow(12, 155, 150, 30);
    delay(500);
    servoSlow(14, 50, 76, 30);
    delay(500);
    servoSlow(10, 105, 20, 30);
    delay(500);
    servoSlow(14, 76, 0, 30);
    delay(500);
    servoSlow(11, 70, 0, 30);
    delay(500);
    servoSlow(13, 0, 15, 30);
    delay(500);
    servoSlow(15, 0, 160, 30);
```

```
    delay(500);
    servoSlow(14, 0, 50, 30);
    delay(500);
    servoSlow(12, 150, 150, 30);
    delay(500);
    servoSlow(14, 50, 90, 30);
    delay(500);
    servoSlow(10, 20, 80, 30);
    delay(500);
    servoSlow(14, 90, 40, 30);
    delay(500);
    servoSlow(10, 80, 0, 30);
    delay(500);
    servoSlow(11, 0, 70, 30);
    delay(500);


     servoSlow(15, 160, 50, 30);
     delay(500);
     servoSlow(14, 40, 0, 30);
     delay(500);
     servoSlow(13, 15, 0, 30);
     delay(500);
     servoSlow(12, 150, 130, 30);
     delay(500);
     servoSlow(11, 70, 85, 30);
     delay(500);
     servoSlow(10, 0, 0, 30);
     delay(500);

  }

 void servoSlow( int num, int start, int pos, int time)  {
  MSS.servoWrite(num, start);
  if(start < pos){
    for ( int i=start; i < pos; i++){

         MSS.servoWrite(num, i);
         delay(time);
      }
      }
       if(start > pos){
        for ( int i= start; i > pos; i--){

         MSS.servoWrite(num, i);
         delay(time);
      }
    }
  }
```

Представлена вище програма реалізує неперервний повторюваний алгоритм. В ході виконання даного алгоритму проводиться послідовне налаштування положень кожного з сервоприводів, з метою досягнення поставленої задачі, а саме: маючи початкове положення необхідно створити шлях по якому б маніпулятор підняв об’єкт з однієї точки і переніс би його в іншу.

Слід відмітити, що в процесі написання вихідного коду була створена невелика функція **servoSlow().** Приймає вона 4 параметри: номер виходу, початкове положення, кінцеве положення, час затримки. Основна задача цієї функції це реалізація можливості плавного повороту сервоприводу, оскільки використання вбудованих в бібліотеку функції не надає такої можливості.

Функція **servoSlow()** працює наступним чином, замість того щоб одразу передати значення кінцевого кутового положення у функцію **servoWrite()**, вона передає послідовно значення від початкового до кінцевого з кроком в один градус та додає затримку після кожної такої ітерації, тим самим реалізуючи плавність повороту.

3.3. Побудова алгоритму з використанням LabView

На базі створеного віртуального приладу для реалізації ручного керування маніпулятором, був доданий функціонал для виконання автоматичного, завчасно визначеного, алгоритму.

Варто відмітити, що побудова алгоритму в середовищі Labview відбувається не тільки в ньому, насправді код Arduino тут також використовується, однак його мета в даній ситуації дещо інша ніж у попередньому випадку. Тут за допомогою коду реалізується функціонал для реалізації можливості використання середовища LabView для керування маніпулятором.

Оскільки середовище для створення алгоритму руху, в даному випадку, являється, здебільшого, графічним то і підхід для реалізації поставленої задачі дещо відрізняється.

В першу чергу була створена можливість перемикання між ручним керування та автоматичним, зроблено це з метою усунення можливих проблем у разі втручання під час виконання автоматичного алгоритму, оскільки досить не раціонально залишати можливість мануального керування маніпулятором в автоматичному режимі, так як це може призвести, в кращому випадку до незначних відхилено від заданого маршруту, або ж, в гіршому випадку, до зупинки проходження маршруту.

Після реалізації перемикання постала задача реалізації самого алгоритму. Як і у випадку з кодом Arduino, керування сервоприводама майже нічим не відрізняється, також маємо номер приводу та кут його повороту.

Для задання послідовності виконання було прийнято рішення реалізувати це наступним чином: була створена матриця розмірності $2 \times n$, де n відповідає числу операцій керування сервоприводами, тобто, якщо алгоритм був спланований таким чином, що в ньому маємо, наприклад, 4 оберти, то відповідно матриця матиме розмірність 2×4. Перший, або, якщо брати з точки зору програмування, нульовий рядок матриці відповідає номеру сервоприводу, в той же час в другому, або ж першому, рядку записані значення кутів. Тобто, якщо зчитувати матрицю по стовпчикам то ми отримуємо два числа, перше, яке відповідає номеру сервопривода, та друге, яке відповідає куту, до якого необхідно буде повернути його.

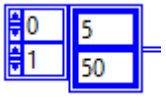

Рисунок 3.3 – Матриця значень.

Методика переходу від одного кутового положення до іншого, по суті, являється досить схожою на ту, що використовувалася при написанні коду в Arduino, тут також обертання відбувається покроково через 1 градус. Для контролю за процесом виконання створена змінна **index (col)**, яка тримає в собі значення поточної колонки, що виконується. На рисунку 3.4 представлена схема відпрацювання повороту та переходу на іншу ітерацію.

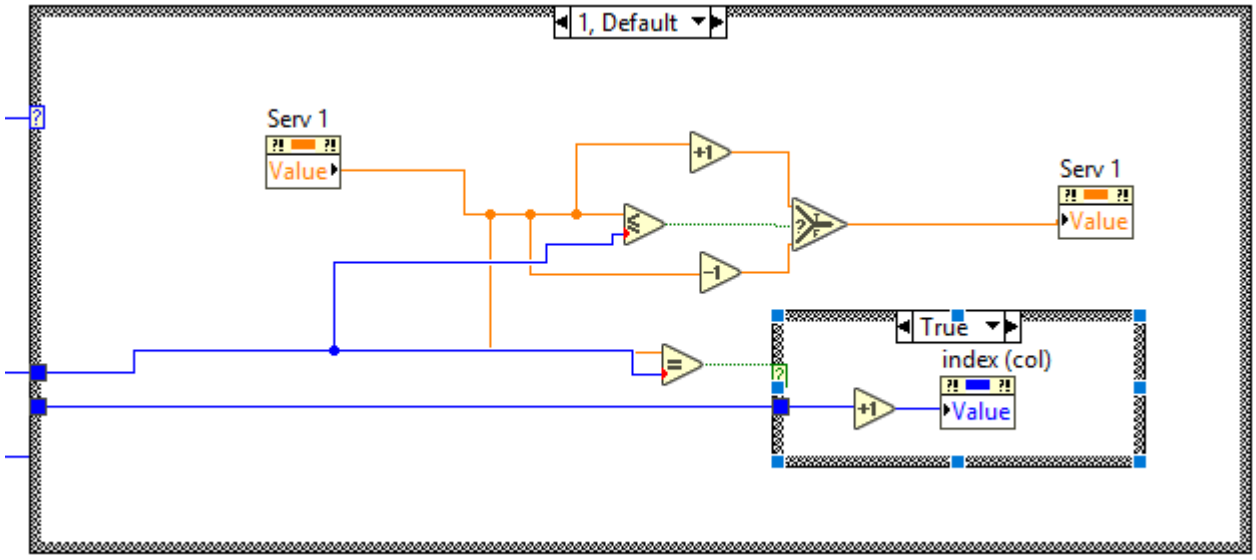

Рисунок 3.4 – Реалізація ітераційного виконання повороту та умови переходу на наступний крок алгоритму.

З метою реалізації постійного виконання заданого алгоритму, створена умова онулення, згідно якої коли значення змінної **index (col)** вийде за межі останнього рядка, проводиться онулення її значення, тим самим алгоритм знову почне виконуватися.

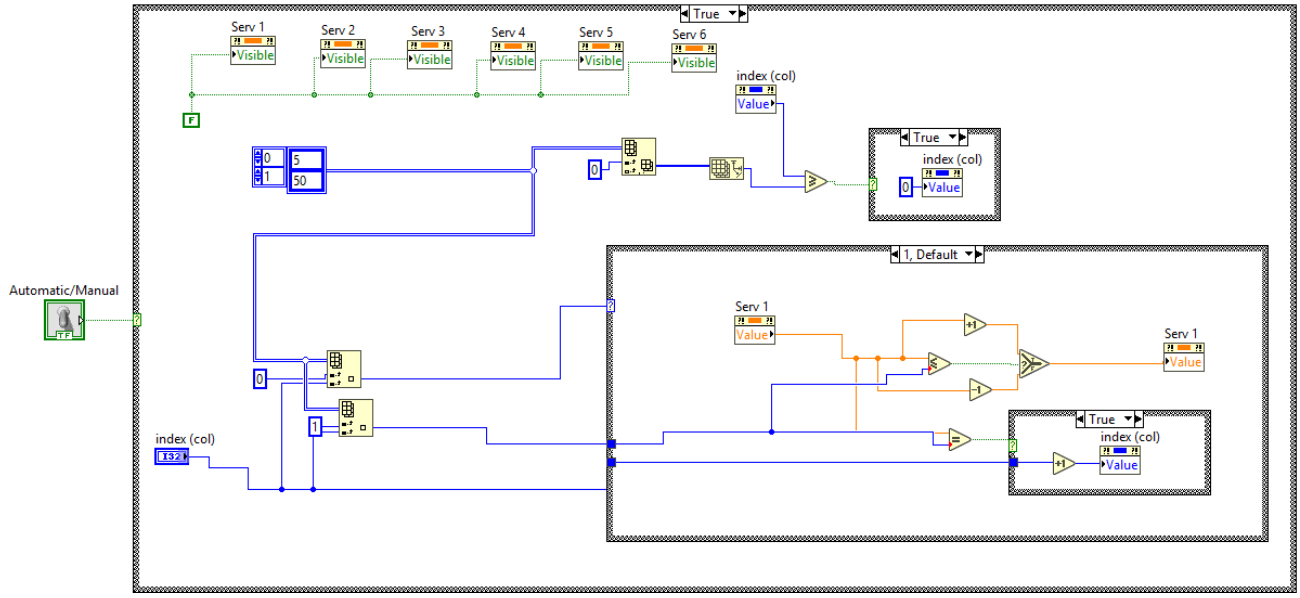

Рисунок 3.5 – Схема для реалізації неперервного та повторного виконання алгоритму в середовищі LabView.

Висновок

В даному дипломному проекті було проведено дослідження промислових роботів.

У якості першого розділу зроблений оглядовий опис складових частин маніпуляторів, та приведений опис найбільш розповсюджених алгоритмів управління. Окрім цього приведено короткі відомості про розвиток технологій робототехніки та перераховані покоління роботів. Також були описані методики за якими проводиться створення алгоритмів керування маніпуляторами.

В другому розділі був проведений аналіз кінематики. Вже на початку створення кінематики стає очевидним, що схема маніпулятора не являється простою, адже в ній присутні не тільки кутові зміщення, а ще і лінійні. Звісно якби проводився огляд лише кінематик кожного окремого сервопривода, то задача була значно легшою, однак такий підхід не дозволив би провести повний комплексний аналіз. На основі методики Денавіта-Хартенберга були визначені необхідні параметри для побудови кінематики, та вирішення прямої та зворотної задачі кінематики. Варто також відмітити що дана методики була використана враховуючи факт того, що вона дозволяє зменшити кількість необхідних координат до чотирьох.

Третій розділ був присвячений здебільшого опис методики створення алгоритму, а також коротко описана динаміка маніпулятора у вигляді рівнянь динаміки та передатних функцій. Основна увага в даному розділі була приділена саме алгоритмам написаним з використанням можливостей середовищ LabView та Arduino. В ході написання програми були перераховані основні можливості, котрі надають середовища для виконання поставленої задачі. Найбільш оптимальним варіантом реалізації поставленої задачі являється комбіноване використання структурного програмування Arduino та графічних методів LabView.

Список використаної літератури

1. Спыну Г. А. Промышленные роботы: конструирование и применение [Текст]/ Спыну Г. А. — Киев: Вища школа, 1985. — 176 с.

2. ГОСТ 25686-85 – Манипуляторы, автооператоры и промышленные роботы. Термины и определения [Текст]. – Москва: Изд-во стандартов, 1988. – 8 с.

3. Спыну Г. А. Промышленные роботы: конструирование и применение 2-е издание[Текст]/ Спыну Г. А. — Киев: Вища школа, 1991. — 307 с.

4. Зенкевич С. Л., Ющенко А. С. Основы управления манипуляционными роботами. 2-е изд. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2004. — 480 с.

5. Корецкий А. В. – Обратная задача кинематики и прямая задача динамики о вертикальном подъёме груза четырёхзвенным манипулятором [Текст]/ Корецкий А. В., Созинова Е. Л. — Москва.: ИНФРА-М, 2015. — 120 с.

6. Иванов А. А. Основы робототехники. 2-е изд. — Москва: ИНФРА-М, 2017. — 223 с.

7. Зенкевич С. Л., Ющенко А. С. Основы управления манипуляционными роботами. 2-е изд. — Москва: Изд-во МГТУ им. Н. Э. Баумана, 2004. — 480 с.

8. Грувер М., Зиммерс Э. САПР и автоматизация производства. — Москва: Мир, 1987. — 528 с.

9. Бурдаков С. Ф., Дьяченко В. А., Тимофеев А. Н. Проектирование манипуляторов промышленных роботов и робототизированных комплексов. — Москва: Высшая школа, 1986. — 264 с. — С

10. Макаров И. М., Топчеев Ю. И. Робототехника: История и перспективы. — Москва: Наука; Изд-во МАИ, 2003. — 349 с.

11. Шахинпур М. Курс робототехники / Пер. с англ. — М.: Мир, 1990. — 527 с.

12. Медведев В. С., Лесков А. Г., Ющенко А. С. Системы управления манипуляционных роботов. — М.: Наука, 1978. — 416 с.

13. Попов Е. П., Верещагин А. Ф., Зенкевич С. Л. Манипуляционные роботы: динамика и алгоритмы. — М.: Наука, 1978. — 400 с.

14. Погорелов А.Д. – Обзор алгоритмов планирования траектории движения манипуляторов

15. Kavraki L.E., Švestka P., Latombe J.C., Overmars M.H., Probabilistic roadmaps for path planning in high-dimensional configuration spaces // IEEE Trans. on Robotics and Automation. 1996. Vol. 12. P. 566–580. DOI: 10.1109/70.508439.

16. Geraerts R., Overmars M., Sampling Techniques for Probabilistic Roadmap Planners // Int. Conference on Intelligent Autonomous Systems (IAS-8), Amsterdam. 2004. P. 600-609.

17. Nissoux C., Siméon T., Laumond J.P., Visibility based probabilistic roadmaps // IEEE Int. Conf. on Intelligent Robots and Systems, Kyongju. 1999. P. 1316–1321.

18. Bohlin R., Kavraki L., Path planning using lazy PRM // IEEE Int. Conf. on Robotic and Automation, San Francisco. 2000. P. 521-528.

19. Dobson A., Krontiris A., Bekris K., Sparse Roadmap Spanners // Algorithmic Foundations of Robotics X, Springer-Verlag Berlin Heidelberg. 2013. P. 279-296. DOI: 10.1007/978-3-642-36279-8.

20. Kuffner J., LaValle S., RRT-connect: An efficient approach to single-query path planning // IEEE Intl. Conf. on Robotics and Automation. 2000. P. 995–1001. DOI: 10.1109/ROBOT.2000.844730.